МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №2**

**з дисципліни «Об’єктно-орієнтоване програмування»**

**на тему: «Перевантаження операцій»**

Виконав: студент гр. ПЗ2011

Кулик Сергій Вадимович

Прийняла:

Демидович Інна Миколаївна

Дніпро, 2021

**Тема:** Перевантаження операцій.

**Мета:** Отримати навики перевантаження операцій у С++.

**Завдання**

Написати програму, що включає реалізацію класу відповідно варіанту індивідуального завдання, а також реалізацію головної функції програми для демонстрація роботи з об’єктами класу.

Клас має включати:

* не менше трьох атрибутів;
* конструктор за замовченням;
* конструктор з параметрами;
* конструктор копіювання;
* деструктор;
* метод для виведення на екран значення всіх атрибутів класу;
* перевантажені операції відповідно до варіанту.

У головній програмі створити меню для вибору операцій. При виконанні операцій на екрані повинні відображатися найменування операції (суть), операнди та результат.

**Індивідуальне завдання**

|  |  |  |
| --- | --- | --- |
| **Варіант** | **Клас** | **Операції** |
| 1 | Автомобіль | Не дорівнює за назвами (!=)  Більше за «пробігом» (>)  Додавання «пробігу» (+) |

**Опис класу**

Клас: автомобіль.

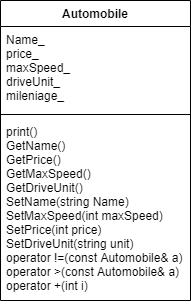
Приватні поля:

* Name\_ ­– ім’я автомобіля.
* price\_ – ціна автомобіля.
* maxSpeed\_ – максимальна швидкість.
* driveUnit\_ – привід автомобіля.
* mileniage – пробіг автомобіля

Публічні методи:

* GetName() – дозволяє отримати ім’я автомобіля.
* GetPrice() – дозволяє отримати ціну автомобіля.
* GetMaxSpeed() – дозволяє отримати макс.швидкість автомобіля.
* GetDriveUnit() – дозволяє отримати тип приводу автомобіля.
* SetName(string Name) – дозволяє установити ім’я автомобіля.
* SetMaxSpeed(int maxSpeed) – дозволяє установити макс.швидкість автомобіля.
* SetPrice(int price) – дозволяє установити ціну автомобіля.
* SetDriveUnit(string unit) – дозволяє установити тип приводу автомобіля.
* print() – виводить інформацію про всі поля класу.
* operator !=(const Automobile& a) – не дорівнює за назвами
* operator >(const Automobile& a) – більше за «пробігом»
* operator +(int i) – додавання «пробігу»

**Діаграмма класів**

****

**Тести до програми**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Назва тесту** | **Вхідні дані** | **Вихідні дані** |
| **1** | Оператор + | b("Honda", 10000, 200, "задний", 20000)  b + 5000 | Mileniage = 25000 |
| **2** | Оператор != | a,  b("Honda", 10000, 200, "задний", 20000) | true |
| **3** | Оператор > | a,  b("Honda", 10000, 200, "задний", 20000) | true |

**Текст програми**

#include <iostream>

using namespace std;

class Automobile

{

public:

Automobile(string Name\_, int price\_, double maxSpeed\_, string driveUnit\_, int mileage\_);

Automobile();

~Automobile();

Automobile(const Automobile& car);

void print();

string GetName() { return Name\_; }

int GetPrice() { return price\_; }

double GetMaxSpeed() { return maxSpeed\_; }

string GetDriveUnit() { return driveUnit\_; }

int GetMileage() { return mileage\_; }

void SetName(string Name) { Name\_ = Name; }

void SetMaxSpeed(int maxSpeed) { maxSpeed\_ = maxSpeed; }

void SetPrice(int price) { price\_ = price; }

void SetDriveUnit(string unit) { driveUnit\_ = unit; }

bool operator !=(const Automobile& a)

{

if (Name\_ != a.Name\_)

{

return true;

}

else

{

return false;

}

}

bool operator >(const Automobile& a)

{

if (mileage\_ > a.mileage\_)

{

return true;

}

else

{

return false;

}

}

Automobile operator +(int i)

{

mileage\_ += i;

return \*this;

}

private:

string Name\_;

int price\_;

double maxSpeed\_;

string driveUnit\_;

int mileage\_;

};

Automobile::Automobile()

{

Name\_ = "Automobile";

price\_ = 0;

maxSpeed\_ = 0;

driveUnit\_ = "Full";

mileage\_ = 0;

}

Automobile::Automobile(string Name\_, int price\_, double maxSpeed\_, string driveUnit\_, int mileage\_)

{

this->Name\_ = Name\_;

this->price\_ = price\_;

this->maxSpeed\_ = maxSpeed\_;

this->driveUnit\_ = driveUnit\_;

this->mileage\_ = mileage\_;

}

Automobile::Automobile(const Automobile& car)

{

Name\_ = car.Name\_;

price\_ = car.price\_;

maxSpeed\_ = car.maxSpeed\_;

driveUnit\_ = car.driveUnit\_;

mileage\_ = car.mileage\_;

}

Automobile::~Automobile()

{

}

void Automobile::print()

{

cout << "Название автомобиля: " << Name\_ << endl;

cout << "Цена: " << price\_ << endl;

cout << "Максимальная скорость: " << maxSpeed\_ << endl;

cout << "Пробег: " << mileage\_ << endl;

cout << "Привод: " << driveUnit\_ << endl << endl;

}

int main()

{

setlocale(LC\_ALL, "ru");

Automobile a, b("Honda", 10000, 200, "задний", 20000);

cout << "Операнди а та b. Пробіг до присваювання: " << b.GetMileage() << endl;

b + 5000;

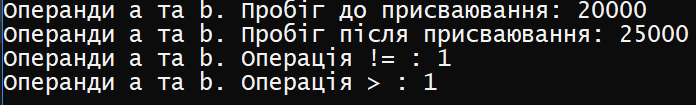
cout << "Операнди а та b. Пробіг після присваювання: " << b.GetMileage() << endl;

cout << "Операнди а та b. Операція != : " << (a != b) << endl; //true

cout << "Операнди а та b. Операція > : " << (b > a) << endl; //true

}

**Результати**



**Висновок**

Під час виконання лабораторної роботи перевантажував опшератори в мові С++. Перевантаження відбувається за допомогою ключового слова operator. Перевантаження операторів необхідне, щоб дати компілятору знати як працювати з нестандартними типами даних.